A Code

Code and demonstrations are in https://github.com/HeegerGao/DMIL.

B Algorithm

Algorithm 1 Dual Meta Imitation Learning

Require: task distribution $p(\mathcal{T})$, multi-task demonstrations $\{\mathcal{D}_i\}, i = 1, \cdots, m$, initial parameters of high-level network θ_h and sub-skill policies $\theta_{l1}, \cdots, \theta_{lK}$, inner and outer learning rate α, β.
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Sample $\{\tau_{i1}\}, \{\tau_{i2}\}, \{\tau_{i3}\}, \{\tau_{i4}\}$ from $\{\mathcal{D}_i\}$
    Evaluate $\nabla_{\theta_h} \mathcal{L}_h(\theta_h, \tau_{i1})$ according to 3 and τ_{i1}
    Compute adapted parameters of high-level network: $\lambda_h = \theta_h - \alpha \nabla_{\theta_h} \mathcal{L}_h(\theta_h, \tau_{i1})$
    Evaluate $\nabla_{\theta_{lk}} \mathcal{L}_{BC}(\theta_{lk}, \mathcal{D}_{2k})$ according to 4 and τ_{i2}, $k = 1, \cdots, K$
    Compute adapted parameters of sub-skills: $\lambda_{lk} = \theta_{lk} - \alpha \nabla_{\theta_{lk}} \mathcal{L}_{BC}(\theta_{lk}, \mathcal{D}_{2k})$, $k = 1, \cdots, K$
    Evaluate $\nabla_{\theta_h} \mathcal{L}_{\mathcal{T}_i}(\lambda_h, \tau_{i3})$ and $\nabla_{\theta_{lk}} \mathcal{L}_{\mathcal{T}_i}(\lambda_{lk}, \mathcal{D}_{4k})$, $k = 1, \cdots, K$
  end for
  Update $\theta_h \leftarrow \theta_h - \beta \nabla_{\theta_h} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(\lambda_h, \tau_{i3})$
  Update $\theta_{lk} \leftarrow \theta_{lk} - \beta \nabla_{\theta_{lk}} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(\lambda_{lk}, \mathcal{D}_{4k})$, $k = 1, \cdots, K$
end while

C Auxiliary Loss

We adopt an auxiliary loss for DMIL to better drive out meaningful sub-skills by punishing excessive switching of sub-skills along the trajectory. This comes from an intuitive idea: each sub-skill should be a temporal-extended macro-action, and the high-level policy only needs to switch to different skills few times along a task, as the same idea of *macro-action* in MLSH [45]. We denote:

$$sign(x) = \begin{cases} 1, \text{ if } x = True \\ 0, \text{ if} x = False \end{cases}, \tag{14}$$

and the auxiliary loss is:

$$\mathcal{L}_{aux}(\tau) = \sum_{t=0}^{T-1} sign(\hat{z}_{t+1} \neq \hat{z}_t) \ / \ len(\tau). \tag{15}$$

Although this operation seems discrete, in practice we can use the operations in modern deep learning framework such as PyTorch [51] to make it differentiable. We add this loss function to the $\mathcal{L}_h(\theta_h, \tau_{i1})$ and $\nabla_{\theta_h} \mathcal{L}(\lambda_h, \tau_{i3})$ with a coefficient $\lambda = 1$. We also perform ablation studies of $\mathcal{L}_{aux}$ and results are in table 7.

D Proofs

D.1 Proof of Lemma 1

Lemma 1 In case $q(\phi_i; \lambda_i)$ is a Dirac-delta function and choosing Gaussian prior for $p(\phi_i|\theta)$, equation 8 equals to the inner-update step of MAML, that is, maximizing $\log p(\mathcal{D}_i^{tr})$ w.r.t. λ_i by early-stopping gradient-ascent with choosing μ_θ as initial point:

$$\lambda_i(\mathcal{D}_i^{tr}; \theta) = \mu_\theta + \alpha \nabla_\theta \log p\left(\mathcal{D}_i^{tr} | \theta\right) |_{\theta = \mu_\theta}. \tag{16}$$

Proof: in case of the conditions of Lemma 1, we have:

$$\lambda_i(\mathcal{D}_i^{tr}; \theta) = \arg\max_{\lambda_i} [\log p(\mathcal{D}_i^{tr} | \mu_{\lambda_i}) - \left\| \mu_{\lambda_i} - \mu_\theta \right\|^2 / 2\Sigma_\theta^2], \tag{17}$$

As stated in [52], firstly in the case of linear models, early stopping of an iterative gradient descent process of λ equals to the maximum posterior estimation (MAP) [53]. In our case the posterior distribution refers to $q(\phi_i|\lambda_i)$, and MAML is a Bayes process to find the MAP estimate as the point estimate of $\lambda_i(\mathcal{D}_i^{tr};\theta)$. In the nonlinear case, this point estimate is not necessarily the global mode of the posterior, and we can refer to [54] for another implicit posterior distribution over ϕ_i and making the early stopping procedure of MAML acting as priors to get the similar result.

D.2 Proof of Equation 8

Equation 8 can be written as:

$$\begin{aligned}\lambda_i(\mathcal{D}_i^{tr},\theta) &= \arg\min_{\lambda_i} \mathrm{KL}(q(\phi_i;\lambda_i)\|p(\phi_i|\mathcal{D}_i^{tr},\theta)) \\ &= \arg\max_{\lambda_i} E_{q(\phi_i;\lambda_i)}[\log p(\phi_i|\mathcal{D}_i^{tr},\theta) - \log q(\phi_i;\lambda_i)] \\ &= \arg\max_{\lambda_i} E_{q(\phi_i;\lambda_i)}[\log p(\mathcal{D}_i^{tr}|\phi_i)] - \mathrm{KL}(q(\phi_i;\lambda_i)\|p(\phi_i|\theta)),\end{aligned} \tag{18}$$

where in MAML we assume $p(\mathcal{D}_i^{tr}|\phi_i) = p(\mathcal{D}_i^{tr}|\phi_i,\theta)$, and use the joint distribution $p(\mathcal{D}_i^{tr},\phi_i|\theta)$ to replace $p(\phi_i|\mathcal{D}_i^{tr},\theta)$ since we assume that $p(\mathcal{D}_i^{tr})$ subjects to uniform distribution. Thus 8 can be proved.

D.3 Proof of Theorem 1

Theorem 1 In case that $\Sigma_\theta \to 0^+$, i.e., the uncertainty in the global latent variables θ is small, the following equation holds:

$$\nabla_\theta \mathcal{L}(\theta,\lambda_1,\cdots,\lambda_M) = \sum_{i=1}^{M} \nabla_{\lambda_i} \log p(\mathcal{D}_i^{val}|\lambda_i) * \nabla_\theta \lambda_i(\mathcal{D}_i^{tr},\theta). \tag{19}$$

Proof:

$$\begin{aligned}\nabla_\theta \mathcal{L}(\theta,\lambda_1,\cdots,\lambda_M) &\approx \sum_{i=1}^{M} \{\nabla_\theta E_{q(\phi_i;\lambda_i)}[\log p(\mathcal{D}_i,\phi_i|\theta) - \log q(\phi_i;\lambda_i)]\} \\ &= \sum_{i=1}^{M} \nabla_\theta [\log p(\mathcal{D}_i^{val}|\lambda_i(\mathcal{D}_i^{tr},\theta)) - \log p(\lambda_i(\mathcal{D}_i^{tr},\theta)|\theta)] \\ &\approx \sum_{i=1}^{M} \nabla_\theta \log p(\mathcal{D}_i^{val}|\lambda_i(\mathcal{D}_i^{tr},\theta)) \\ &= \sum_{i=1}^{M} \nabla_{\lambda_i} \log p(\mathcal{D}_i^{val}|\lambda_i) * \nabla_\theta \lambda_i(\mathcal{D}_i^{tr},\theta)\end{aligned} \tag{20}$$

where the first approximate equal holds because the VI approximation error is small enough, and the second approximate equal holds because that in case $\Sigma_\theta \to 0^+$ and assuming λ_i be a neuron network, $\log p(\lambda_i(\mathcal{D}_i^{tr},\theta)|\theta) \approx 0$ holds almost everywhere, so $\nabla_\theta \log p(\lambda_i(\mathcal{D}_i^{tr},\theta)|\theta) \approx 0$. Note the condition of theorem 1 is usually satisfied since we are using MAML, and the initial parameters θ are assumed to be deterministic.

From another perspective, the right side of above equation is the widely used meta-gradient in MAML, and it is equal to $\frac{1}{m}\sum_{i=1}^{m}\left(I - \alpha\nabla_\theta^2 \mathcal{L}_{BC}\left(\theta,\mathcal{D}_i^{tr}\right)\right) * \nabla_{\lambda_i}\mathcal{L}_{BC}\left(\theta - \alpha\nabla_\theta \mathcal{L}_{BC}\left(\theta,\mathcal{D}_i^{tr}\right),\mathcal{D}_i^{val}\right)$, which is proved to be converged by [17].

D.4 Proof of Theorem 2

Theorem 2 In case of $p(a_t|s_t,\theta_{lk}) \sim \mathcal{N}(\mu_{\theta_{lk}(s_t)},\sigma^2)$, we have:

$$\nabla_{\theta_h} \log p(\mathcal{D}_i^{tr}|\theta_h,\theta_{l1},\cdots,\theta_{lK}) = \nabla_{\theta_h}\mathcal{L}_h(\theta_h,\mathcal{D}_i^{tr}), \tag{21}$$

and

$$\nabla_{\theta_{lk}} \log p(\mathcal{D}_i^{tr}|\theta_h, \theta_{l1}, \cdots, \theta_{lK}) = \nabla_{\theta_{lk}} \mathcal{L}_{BC}(\theta_{lk}, \mathcal{D}_{2k}), k = 1, \cdots, K. \tag{22}$$

Proof: since $p(\mathcal{D}_i^{tr}|\theta_h, \theta_{l1}, \cdots, \theta_{lK}) = \prod_{t=1}^{N} p(a_t|s_t, \theta_h, \theta_{l1}, \cdots, \theta_{lK}) p(s_t|\theta_h, \theta_{l1}, \cdots, \theta_{lK})$ and the second term is independent of θ, we consider the first conditional probability:

$$p(a_t|s_t, \theta_h, \theta_{l1}, \cdots, \theta_{lK}) = \sum_{k=1}^{K} p(z_k|s_t, \theta_h) p(a_t|s_t, \theta_{lk}). \tag{23}$$

In the HI step, $p(a_t|s_t, \theta_{lk})$ is fixed, thus 23 becomes a convex optimization problem:

$$\max_{\theta_h} \sum_{k=1}^{K} p(z_k|s_t, \theta_h) p(a_t|s_t, \theta \tag{24}$$

The solution of this problem is λ_h^* which satisfies $p(z_k|s_t, a_t, \lambda_h^*) = 1$, $k = \arg\max_k p(a_t|s_t, \theta_{lk})$. This means that π_{θ_h} needs to predict the sub-skill category at time step t as k, in which case $\pi_{\theta_{lk}}$ can maximize $p(a_t|s_t, \theta_{lk})$. In case we choose π_{θ_h} to be a classifier that employs a Softmax layer at the end, minimizing the cross entropy loss 3 equals to maximize 24, thus 21 can be proved.

In the LI step, $p(z_k|s_t, \lambda_h)$ is fixed, and the data sets for optimizing $\theta_{l1}, \cdots, \theta_{lK}$ are also fixed as $\mathcal{D}_{2k} = \{(s_{ijt}, a_{ijt})|\hat{z_{i2t}} = k\}_{t=1}^{N_k}$. Thus we need to maximize each $p(a_t|s_t, \theta_{lk})$ with $\mathcal{D}_{2k}$. In case of $p(a_t|s_t, \theta_{lk}) \sim \mathcal{N}(\mu_{\theta_{lk}(s_t)}, \sigma^2) \propto \exp[-\frac{(a_t - \pi_{\theta_{lk}}(s_t))^2}{2\sigma^2}]$, we have $\max_{\theta_{lk}} p(a_t|s_t, \theta_{lk}) \Leftrightarrow \min_{\theta_{lk}} (a_t - \pi_{\theta_{lk}}(s_t))^2$, which leads to the loss function 4, thus 22 can be proved, which finishes the prove of Theorem 2.

D.5 Proof of the E-step of DMIL

According to Theorem 1, we aim to maximize 17 w.r.t λ_i from the initial point θ_i with coordinate gradient ascent. We here need to prove that in DMIL, we could also achieve the global maximum point of λ_i as in MAML. We first give out the following Lemma:

Lemma 2 Let x be the solution found by coordinate gradient descent of $f(x)$. Let $x_i, i = 1, \cdots, n$ be the n coordinate directions used in the optimization process. If $f(x)$ can be decomposed as:

$$f(x) = g(x) + \sum_{i=1}^{n} h_i(x), \tag{25}$$

where $g(x)$ is a differentiable convex function, and each $h_i(x)$ is a convex function of the coordinate direction x_i, then x is the global minimum of $f(x)$.

Proof: Let y be another arbitrary point, we have:

$$\begin{aligned} f(y) - f(x) &= g(y) + h(y) - (g(x) + h(x)) \\ &\geq \nabla_x g(x)^T (y - x) + \sum_{i=1}^{n} h_i(y_i) - h_i(x_i) \\ &= \sum_{i=1}^{n} (\nabla_i g(x)(y_i - x_i) + h_i(y_i) - h_i(x_i)) \\ &\geq 0. \end{aligned} \tag{26}$$

Now let's consider our problem. Consider

$$\begin{aligned}
\log p(\mathcal{D}_i^{tr}|\theta_{ih},\theta_{il}) &= \log \sum_{t=1}^{T} p(a_t|s_t,\theta_{ih},\theta_{il})p(s_t) \\
&= \log \sum_{t=1}^{T} p(s_t) \sum_{k=1}^{K} p(z_k|s_t,a_t,\theta_{ih})p(a_t|s_t,\theta_{il}) \\
&\geq \sum_{t=1}^{T} [\log p(s_t) + \log \sum_{k=1}^{K} p(z_k|s_t,a_t,\theta_{ih})p(a_t|s_t,\theta_{il})] \\
&\geq \sum_{t=1}^{T} [\log p(s_t) + \sum_{k=1}^{K} \log p(z_k|s_t,a_t,\theta_{ih})p(a_t|s_t,\theta_{il})] \\
&= \sum_{t=1}^{T} [\log p(s_t) + \sum_{k=1}^{K} \log p(z_k|s_t,a_t,\theta_{ih}) + \sum_{k=1}^{K} \log p(a_t|s_t,\theta_{il})].
\end{aligned} \tag{27}$$

In our case, two coordinate directions are θ_{ih} and θ_{il}. Let's consider the terms inside the brackets. According to Lemma 2, we can think $\log p(s_t)$ as $g(x)$(here it equals to constant), $\sum_{k=1}^{K} \log p(z_k|s_t,a_t,\theta_{ih})$ as $h_1(x)$ and $\sum_{k=1}^{K} \log p(a_t|s_t,\theta_{il})$ as $h_2(x)$. Thus the optimum can be proved.

E Additional Ablation Studies

E.1 Effects of the Bi-level Meta-learning Process

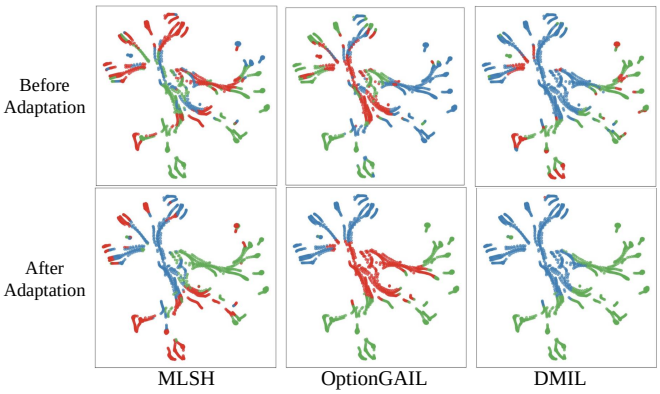
We use two variants of DMIL to see the effectiveness of bi-level meta-earning process. **DMIL-High**: a variant that only meta-learns the high-level network, and **DMIL-Low**: a variant that only meta-learns sub-skills. We use Option-GAIL as a comparison that does not meta-learn any level of the hierarchical structure.

table 4 shows the results of this ablation study. DMIL-High achieves close results with OptionGAIL in all meta-training suites and better results in all meta-testing suites, but worse than DMIL in all cases. This shows that meta-learning the high-level network can help the hierarchical structure adapt to new tasks, but only transferring the high-level network is not enough for accomplish all kinds of new tasks. DMIL-Low achieves poor results in all suites except in ML10 meta-training suites. This shows that transferring the high-level network is necessary when training on a large scale of tasks or testing in new tasks.

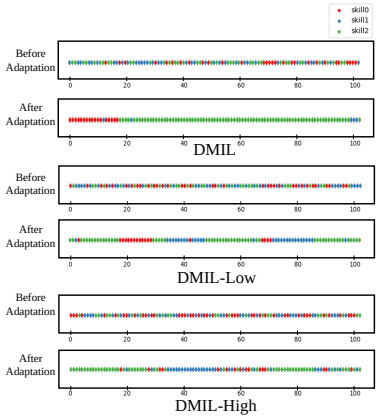
Table 4: Success rates of DMIL-High, DMIL-Low, DMIL and OptionGAIL on Meta-world environments with $K = 3$. Each data point comes from the success rate of 20 tests.

	ML10				ML45			
	Meta-training		Meta-testing		Meta-training		Meta-testing	
Methods	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot
OptionGAIL	0.755±0.011	**0.952±0.016**	0.241±0.042	0.640±0.025	0.506±0.008	0.715±0.006	0.220±0.013	0.481±0.010
DMIL-High	0.634±0.001	0.914±0.011	0.298±0.012	0.670±0.015	0.495±0.007	0.735±0.006	0.280±0.016	0.551±0.011
DMIL-Low	0.746±0.011	0.943±0.006	0.291±0.024	0.666±0.021	0.511±0.005	0.765±0.009	0.266±0.010	0.492±0.014
DMIL	**0.775±0.010**	0.949±0.009	**0.396±0.016**	**0.710±0.021**	**0.590±0.010**	**0.859±0.008**	**0.376±0.004**	**0.640±0.009**

For better understand the effect of bi-level meta-learning process, we perform ablation study for three DMIL variants in an manually-designed new task *push-around-wall* (fig. 8). In this task, the robot needs to grasp a cube and circle it around the wall. This is a brand new skill that is not in the meta-world suite. The accomplishment of this new task requires quickly adapting abilities of both the high-level network and sub-skills. We sample two demonstrations and use the first one as few-shot data, and illustrate the sub-skill categories of the second demonstration given by the high-level network at fig. 4(b). Before adaptation, all variants give out approximately random results. However, after one-shot adaptation, DMIL classifies almost every state into sub-skill 0 and sub-skill 2, which indicates these two sub-skills in DMIL have been adapted to the new task, and the high-level



(a) T-sne results of demonstrations and sub-skill categories of several hierarchical models for meta-testing task *hand-insert*.



(b) Sub-skill categories of task *push-around-wall* for ablation study.

Figure 4: T-sne results and ablation studies about the bi-level meta-learning process.

network has also been adapted with the supervision from adapted sub-skills. Compared to DMIL, DMIL-Low still can not give out reasonable results after adaptation, since its high-level network lacks the ability to quickly transfer to new tasks. Instead, DMIL-High gives out plausible results after adaptation. This shows the high-level network has adapted to the new task according to the supervision from adapted sub-skills, but no sub-skill can dominate for a long time period since all sub-skills lack the quickly adaptation ability.

E.2 T-sne Results of Different Methods

For comparison of different methods, we illustrate the t-sne results of states of each sub-skill in an ML45 meta-testing task hand-insert in fig. 4(a). We use 3 demonstrations for adaptation, and draw the t-sne results on another 16 demonstrations. This task is a meta-testing task, so no method has ever been trained on this task before.

MLSH shows almost random clustering results no matter before and after adaptation, since its high-level network is relearned in new tasks. OptionGAIL clusters to three sub-skills after adaptation. Compared to them, DMIL clusters the data to only two sub-skills after adaptation. We believe fewer categories reflect more meaningful sub-skills are developed in DMIL.

E.3 Effects of Sub-skill Number K in The Kitchen Environment

We perform ablation studies of sub-skill number K on the Kitchen environments and choose $K = 2, 4, 8$ respectively. table 5 shows the results. We can see that a smaller number of sub-skills can

achieve better results on such four unseen results that a large number of sub-skills. This may indicate that the sub-skill number K can work as a 'bottleneck' like the middle layer in an auto-encoder.

Table 5: Ablations of sub-skill number K in Kitchen environments.

Task (Unseen)	K=2	K=4	K=8
Microwave, Kettle, Top Burner, Light Switch	**1.9±0.43**	1.5±0.48	1.7±0.22
Microwave, Bottom Burner, Light Switch, Slide Cabinet	2.15±0.19	**2.35±0.39**	2.0±0.37
Microwave, Kettle, Hinge Cabinet, Slide Cabinet	2.45±0.25	**3.15±0.22**	1.85±0.23
Microwave, Kettle, Hinge Cabinet, Slide Cabinet	2.01±0.24	**2.95±0.44**	2.44±0.47

E.4 Effects of Fine-tuning Steps

As all few-shot learning problems, the fine-tuning steps in new tasks to some extent determine the performance of the trained model. It controls the balance between under-fitting and over-fitting. We perform ablation studies of fine-tune steps in Meta-world benchmarks with $K = 5$ and lr=1e-2 in table 6. Results are as follows:

Table 6: Ablation studies of the fine-tuning steps in Meat-world experiments with $K = 5$.

	ML10				ML45			
	Meta-training		Meta-testing		Meta-training		Meta-testing	
fine-tune steps	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot
10	0.5	0.575	**0.28**	0.24	0.374	0.49	0.05	0.17
30	0.69	**0.915**	0.25	0.31	0.602	0.85	0.13	0.32
50	**0.695**	0.895	0.27	0.39	0.583	0.87	0.12	0.32
100	0.665	0.905	0.23	0.41	0.614	0.872	0.07	0.43
300	0.66	0.845	**0.28**	**0.44**	**0.605**	**0.876**	0.12	**0.44**
500	0.63	0.91	0.25	0.39	0.584	0.867	**0.13**	0.42
Range	0.195	0.34	0.05	0.2	0.231	0.386	0.08	0.27

E.5 Effects of Continuity Regularization

We perform ablation studies of the effect of continuity regularization as following table 7. DMIL_nc means no continuity regularization. We can see that the continuity constraint would damage meta-training performance slightly, but increase the meta-testing performance greatly.

Table 7: $K = 10$

	ML10				ML45			
	Meta-training		Meta-testing		Meta-training		Meta-testing	
variants	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot
DMIL	**0.795**	0.94	**0.52**	**0.57**	**0.713**	0.92	**0.21**	**0.48**
DMIL_nc	0.788	**0.96**	0.32	0.56	0.703	**0.927**	0.17	0.35
Gap	0.007	-0.02	0.2	0.01	0.01	-0.007	0.04	0.13

E.6 Effects of Hard/Soft EM Choices

In DMIL, we use hard EM algorithm to train the high-level network. One may think about to use soft cross entropy loss to train the high-level network to get better results. We perform this ablation study in the following table 8. We can see that a soft cross entropy training won't help increase the whole success rates. This may comes from that, usually we use soft cross entropy (such as label smoothing) to prevent over-fitting. However, in our situation, this may cause under-fitting, since training on such

Table 8: Ablation about hard/soft EM choices with $K = 5$ in the Meta-world environments.

	ML10				ML45			
	Meta-training		Meta-testing		Meta-training		Meta-testing	
variants	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot	1-shot	3-shot
DMIL	**0.795**	**0.94**	**0.52**	**0.57**	**0.713333**	**0.92**	**0.21**	**0.48**
DMIL_soft	0.37	0.65	0.33	0.46	0.235556	0.43	0.1	0.32
Gap	0.425	0.29	0.19	0.11	0.477778	0.49	0.11	0.16

Table 9: DMIL hyper-parameters.

Parameter	Value
α	5e-4
β	1e-4
fine-tune iterations	3
batch size (in trajectory)	16
λ	0.1

a large scale of diverse manipulation tasks is already very difficult. Future works can seek more comparisons about this choice.

Additionally, we found an interesting phenomena that the training loss of the high-level network with a softmax shows a trend of rising first and then falling, as shown in fig. 5. In our experiments, a softmax loss may regularize the optimization process to make the high-level network be under-fitting on training data. This may come from that, the experiment environment (Meta-world) contains a large scale of manipulation tasks, in which the training of the high-level network can be difficult and unstable. Thus a soft-max cross entropy loss cannot help that much here like how it works as a regularizer to prevent over-fitting in the label smoothing [55].

Figure 5: The training loss of the high-level network with a softmax shows a trend of rising first and then falling.

F Experiment Details

F.1 Environments

See fig. 7,fig. 6,fig. 9 and fig. 8.

F.2 Model Setup

DMIL: The high-level network and each sub-skill is modeled with a 4-layer fully-connected neuron network, with 512 ReLU units in each layer. We use Adam as the meta-optimizer. **DMIL-High** and **DMIL-Low** use the same architecture with DMIL. Hyper-parameter settings are available in table 9.

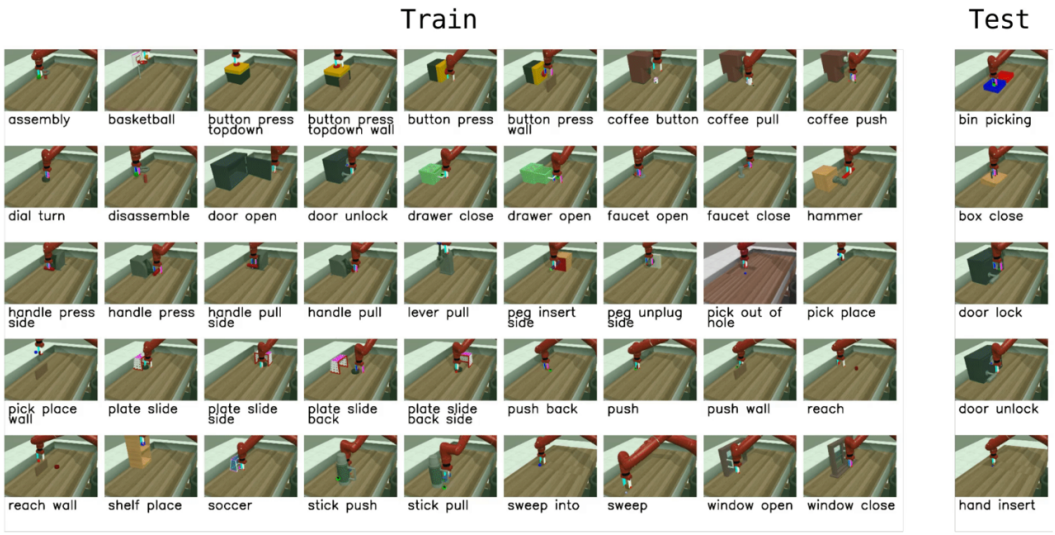


Figure 6: The ML45 environment.

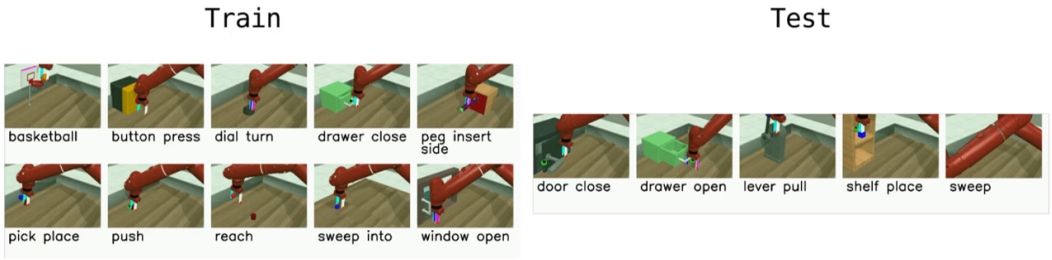


Figure 7: The ML10 environment.

MIL: We use a transformer [50] as the policy to perform MAML. The input of the encoder is the whole one-shot demonstration or 3-shot demonstrations with concatenated state and action. The input of the decoder is current state. The output is the predicted action. Hyper-parameter settings are available in table 10.

Table 10: MIL hyper-parameters.

Parameter	Value
n_{head}	8
n_{layer}	3
d_{model}	512
d_k	64
d_v	64
$n_{position}$	250
dropout	0.1
batch size (in state-action pair)	512

MLSH: We use the same settings of network with DMIL here. The macro step of high-level network is 3. Since our problem is not a reinforcement learning process, we use the *behavior cloning* variant of MLSH. The pseudo reward is defined by the negative mean square loss of the predicted action and the ground truth, and we perform pseudo reinforcement learning process with off-policy demonstration data. We use PPO as our reinforcement learning algorithm. Note in this way we ignore the importance sampling weights that required by replacing the sampling process in the environments with the demonstrations in the replay buffer, which has been shown to be effective in practice in [56, 57]. Hyper-parameter settings are available in table 11.

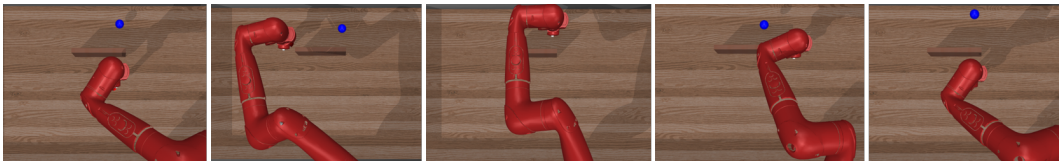


Figure 8: Task *push-around-wall*.

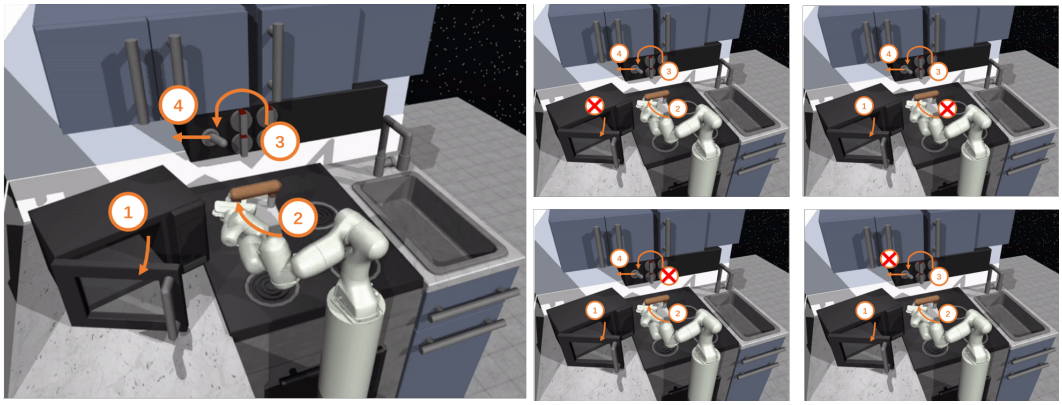


Figure 9: Kitchen environments.

PEMIRL: We use the same setting of the high-level network as the policy π_ω and the inference model q_ψ in PEMIRL. We use PPO as our reinforcement learning algorithm. We use a 3-layer fully-connected neuron network as the context-dependent disentangled reward estimator $r_\theta(s, m)$ and the context-dependent potential function $h_\phi(s, m)$. Here we also use the *behavior cloning* variant of PEMIRL to only train models on the off-policy data. Hyper-parameter settings are available in table 12.

F.3 Training Details

For fine-tuning, OptionGAIL, DMIL-Low and DMIL-High have no meta-learning mechanism for (some parts of) the trained model. In the few-shot adaptation process, we have different fine-tune method for these baselines:

OptionGAIL: We train OptionGAIL models on the provided few-shot demonstrations for a few epochs.

DMIL-Low: We fix the high-level network and only fine-tune sub-skills with few-shot demonstrations.

DMIL-High: We fix sub-skills and only fine-tune the high-level network with few-shot demonstrations.

Table 11: MLSH hyper-parameters.

Parameter	**Value**
high-level learning rate	1e-3
sub-skill learning rate	1e-4
PPO clip threshold	0.02
high-level warmup step	500
joint update step	1000
batch size (in state-action pair)	900

Table 12: PEMIRL hyper-parameters.

Parameter	**Value**
learning rate of all models	1e-4
PPO clip threshold	0.02
coefficient (γ) of h_ϕ	1
β	0.1
batch size (in trajectory)	16